Binary Constrained Deep Hashing Network for Image Retrieval without Manual Annotation

Thanh-Toan Do[†] Tuan Hoang[‡] Dang-Khoa Le Tan[‡] Trung Pham[⋆] Huu Le[⋆]
Ngai-Man Cheung[‡] Ian Reid[⋆]

[†]University of Liverpool, [‡]Singapore University of Technology and Design, [⋆]University of Adelaide

thanh-toan.do@liverpool.ac.uk, nguyenanhtuan_hoang@mymail.sutd.edu.sg,
{letandang_khoa,ngaiman_cheung}@sutd.edu.sg, {trung.pham,huu.le,ian.reid}@adelaide.edu.au

Abstract

Learning compact binary codes for image retrieval task using deep neural networks has attracted increasing attention recently. However, training deep hashing networks for the task is challenging due to the binary constraints on the hash codes, the similarity preserving property, and the requirement for a vast amount of labelled images. To the best of our knowledge, none of the existing methods has tackled all of these challenges completely in a unified framework. In this work, we propose a novel end-to-end deep learning approach for the task, in which the network is trained to produce binary codes directly from image pixels without the need of manual annotation. In particular, to deal with the non-smoothness of binary constraints, we propose a novel pairwise constrained loss function, which simultaneously encodes the distances between pairs of hash codes, and the binary quantization error. In order to train the network with the proposed loss function, we propose an efficient parameter learning algorithm. In addition, to provide similar / dissimilar training images to train the network, we exploit 3D models reconstructed from unlabelled images for automatic generation of enormous training image pairs. The extensive experiments on image retrieval benchmark datasets demonstrate the improvements of the proposed method over the state-of-the-art compact representation methods on the image retrieval problem.

1. Introduction

We are interested in learning compact image representations for large scale content-based image retrieval problem. Recent researches have applied deep learning to image retrieval problem and achieved improvements in comparison to traditional local feature approaches. In [41, 3], the authors show that using the real-valued features from off-the-shelf pretrained networks to represent images achieve impressive retrieval results. In [5, 1, 13] the authors further show that fine-tuning pretrained deep networks for image retrieval task helps to boost the retrieval performance. However, to fine-tune a deep network, it requires an enormous amount of labelled images which is not easy to achieve. It is because annotating images with labels or tags requires skilled manpower, and the label of an image is not always well defined. In addition, although representing images by real-valued high dimensional features from deep networks achieves high retrieval accuracy, it is not applicable for large scale retrieval problem. It is because these representations cause expensive storage and time-consuming searching.

Using binary hash codes to represent images is an attractive approach for large scale vision problems including image retrieval because the binary codes allow the fast computation and efficient storage [6, 12, 58, 27, 32, 8, 15, 56, 34, 9, 30, 31, 57]. However, learning binary codes in deep networks is challenging. This is because one has to deal with the binary constraint on the hash codes. Furthermore, another important requirement of hashing is the similarity preservation, i.e., similar/dissimilar images should have similar/dissimilar binary codes. To achieve this requirement under deep models, previous deep hashing methods require a vast of amount of manually well-defined labelled datasets to supervise the training or fine-tuning. Unfortunately, such large labelled datasets are not always available, especially in some problems which are not directly based on classification such as the image retrieval.

In this paper, we aim to address the above challenges by learning an end-to-end deep neural network to produce binary hash codes directly from images without the need of manually labelled datasets. In particular, we propose a pairwise binary constrained loss function to model the relative similarities between pairs of hash codes and to encourage the network outputs to be binary values. In order to train the

network with the proposed loss, we propose a novel learning scheme that is inspired from the penalty method and alternating optimization. Furthermore, since our loss function is pairwise, it only requires relative relationship for pairs of images, i.e., matching and non-matching images. Clearly such relationship can be obtained without resorting semantic labels. Inspired by the recent works [1, 10] we exploit 3D models built from unlabelled images using Structure-from-Motion (SfM) to automatically create training data. As a result, the training of our deep hashing network can be done completely and automatically by using unlabelled images as inputs.

In summary, we make the following contributions. 1) We propose a novel end-to-end deep learning framework for learning compact binary codes in which the input for training the framework is only unlabelled images. 2) We propose a novel pairwise loss function that simultaneously encodes the distances between hash codes and the binary quantization error. To train the network with the proposed loss, we develop an efficient alternating optimization to optimize the network parameters. 3) We exploit reconstructed 3D models to automatically create the training data. To the best of our knowledge, this work is the first one that relies on 3D geometry to create data for training an end-to-end deep hashing framework. 4) We perform solid experiments on three image retrieval datasets to demonstrate the improvements of the proposed method over the state of the art.

The remainder of the paper is organized as follows: Section 2 discusses related works. Sections 3 and 4 present the proposed method and experimental results, respectively. Section 5 concludes the paper.

2. Related Work

In this section, we review previous works related to the context of our work. Those include traditional hashing approaches, deep hashing approaches, and end-to-end image retrieval with weakly supervised / unsupervised fine-tuning.

Traditional hashing methods: Existing binary hashing methods can be categorized as data-independent and data-dependent schemes [52, 53, 14]. Data-independent hashing methods [11, 24, 40, 25] rely on random projections for constructing hash functions. Although representative data-independent hashing methods such as Locality-Sensitive Hashing (LSH) [11] and its kernelized versions [24, 40] have theoretical guarantees that the more similar data would have higher probability to be mapped into similar binary codes, they require long codes to achieve high precision. Different from data-independent approaches, data-dependent hashing methods use available training data for learning hash functions in unsupervised or supervised manner and they usually achieve better retrieval results than data-independent methods. The unsupervised hashing methods [54, 12, 15, 6, 16] try to preserve the neighbor similarity of samples in Hamming space without semantic label information. The representative unsupervised hashing methods include Iterative Quantization (ITQ) [12], Spherical Hashing (SPH) [16], K-means Hashing (KMH) [15], etc. The supervised hashing methods [23, 33, 29, 44, 36] try to preserve the label similarity of samples using labelled training data. The representative supervised hashing methods include Kernel-Based Supervised Hashing (KSH) [33], Semi-supervised Hashing (SSH) [51], Supervised Discrete Hashing (SDH) [44], Binary Reconstructive Embedding (BRE) [23].

Deep hashing methods: All the previous hashing methods are originally designed and experimented on hand-crafted features which may limit their performance in practical applications. Recently, to leverage the power of deep convolutional neural networks (CNNs) [28, 22, 46], few deep unsupervised hashing and many deep supervised hashing methods have been proposed.

There are few deep models which are proposed for unsupervised hashing [48, 30, 31, 45]. In [30, 31], the authors proposed an end-to-end deep learning framework for unsupervised hashing. The network is trained to produce hash codes that minimize the quantization loss with the output of the last VGG's [46] fully connected layer. In [48], the authors proposed a region-based deep hashing method in which the network consists of three modules, i.e., object proposal generation, feature extraction, and a hashing layer. In [45] the authors proposed an unsupervised deep hashing method that alternatingly proceeds over three training modules: deep hash model training, similarity graph construction, and binary code optimization.

Different from unsupervised setting, there are many methods have been proposed for deep supervised hashing. In [55] the authors proposed a two-step supervised hashing method which learns a deep CNN based hash function with the pre-computed binary codes. Recently, many works proposed end-to-end deep supervised hashing methods in which the image features and the hash codes are simultaneously learned [58, 27, 57, 56, 32, 59]. Most of those models consist of a deep CNN for image feature extraction and a hashing layer that tries to approximate the *sign* function. By joint optimization, the produced hash codes have been shown more sufficient to preserve the semantic similarity between images. Ideally, the hashing layer should adopt a $sign$ activation function to output exactly binary codes. However, due to the vanishing gradient difficulty of the $sign$ function, an approximation procedure must be employed. For example, $sign$ can be approximated by a tanh-like function $y = tanh(\beta x)$, where β is a free parameter controlling the trade off between the smoothness and the binary quantization loss [57]. However, it is non-trivial to determine an optimal β. A small β causes large binary quantization loss while a big β makes the output of

the function close to the binary values, but the gradient of the function almost vanishes, making back-propagation infeasible. The problem still remains when the sigmoid-like functions [58, 27, 56, 59] are used. Another drawback of deep supervised hashing networks is the requirement for a large amount of semantic annotated data which will be used for encoding the semantic similarities in the loss function. However, such large annotated data is usually unavailable in the large scale image retrieval problem.

Weakly supervised / unsupervised fine-tuning: In the last few years, image retrieval has witnessed an increasing of performance due to better image representations. In particular, features obtained from pretrained CNN models, which are trained on image classification task, are usually adopted. For example, [49, 3] used convolutional features, while [41] used fully connected features for the image retrieval. Fine-tuning the pretrained networks has also shown further improvements [5]. However, the fine-tuning requires the availability of annotated data. Unfortunately, it would be difficult and expensive to manually label a large collection of images. To overcome this challenge, the recent works proposed to use weakly supervised or unsupervised fine-tuning. In [1], to prepare the data to fine-tune the network for the place recognition task, the authors used a weakly supervised approach, in which they used Google Street View Time Machine for getting GPS-tagged panoramic images taken at nearby spatial locations on the map. Two images taken far from each other are considered as non-matching, while the matching images are selected by the most similar nearby images. In [39, 10], the authors made further improvements over [1], i.e., discovering matching and non-matching pairs in a totally unsupervised manner. Using a large amount of images downloaded from Flickr with keywords of popular landmarks and cities, they applied Structure from Motion [38] for building multiple 3D models. Images belonging to the same 3D model and sharing enough 3D points are considered as matching, while images belonging to different 3D models are considered as non-matching.

3. Binary Constrained Deep Hashing Network without Manual Annotation

Figure 1 illustrates the proposed pipeline which trains a deep network for learning binary codes without the need of manually annotated data. In the following, we describe the proposed framework including: the network architecture, the automation of training data generation, the pairwise loss function, and the learning of the network parameters.

3.1. Network architecture

The proposed network (Figure 1) comprises of four components: (i) a feature extraction component for extracting image representations; (ii) a fully connected layer for reducing dimension of the image features (Dimensionality Reduction – DR layer). The number of units of this layer equals to the code length required to represent each input image; (iii) a fully connected layer which maps the reduced real-valued features to binary values (Hash Code – HC layer); (iv) a pairwise loss function which acts on the outputs of the HC layer. It is worth noting that the choice of the feature extraction component is flexible in our framework. It can be the standard convolutional neural network architectures, e.g. AlexNet [22] or VGG [46], in which the outputs of their last fully connected layer are used as inputs for the DR layer. Alternatively, the recent architecture which replaces the fully connected layers of VGG or AlexNet by a Maximum Activations of Convolutions (MAC) layer [49] can also be used. In this case, the MAC features will be used as inputs for the DR layer. For the image retrieval task focused in this paper, we adopt the VGG network with a MAC layer as we empirically find that using MAC layer gives better performance than fully connected layer.

Specifically, the MAC layer can be described as follows: Given an input image, the output of the last convolutional layer of VGG is a 3D tensor $W \times H \times K$, where K is the number of output feature maps which equals to 512, and $W \times H$ is spatial size of the last convolutional layer. Let $\mathcal{X}_k$ be k^{th} feature map. The MAC image representation is constructed by

$$\mathbf{u} = [u_1, \ldots, u_k, \ldots, u_K]^T, \text{where } u_k = \max_{x \in \mathcal{X}_k} x \qquad (1)$$

It is worth noting that the previous work [39] also used a pairwise loss for fine-tuning deep network. However, our work significantly differs from [39] at many aspects: while the target in [39] is to learn real-valued representations (i.e., 512-D real-valued features), our work aims to learn compact binary codes. To this end, we have two additional layers, i.e., dimensionality reduction and hash code, which are trained end-to-end together with other components. More important, our pairwise loss (Section 3.3) has binary constraints, unlike constraint-free loss as [39]. The new layers and binary constrained loss function are crucial for hashing, i.e., it ensures the model to produce compact binary codes. Furthemore, we propose a learning scheme to cope with the binary constrained loss function (Section 3.4). This differs from [39] which simply uses standard back-propagation to train the network with the constraint-free loss.

3.2. Training data

The training input for our network is pairs of matching and non-matching images. One way to achieve training pairs is to access to semantic (class) labelled images so that we could train a hash function which returns similar hash codes for images with the same label, or vice versa. Unfortunately, such labelled dataset is not always available, especially for some problems which are not directly based

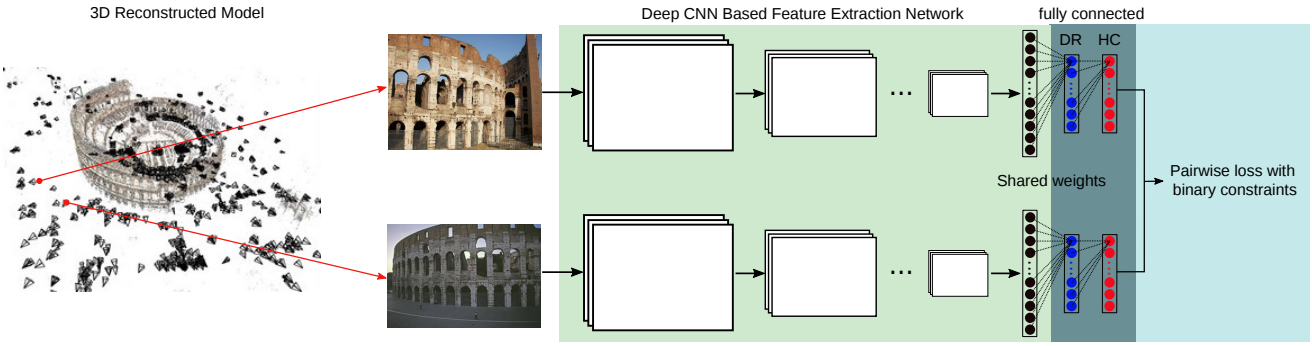


Figure 1: The overview of the proposed deep hashing. Training data is created automatically by exploiting 3D reconstructed models and their associated images. The network architecture comprises of four components: (i) convolutional layers which is followed by a MAC layer (black layer) for extracting image representations; (ii) a fully connected layer for Dimensional Reduction (blue layer); (iii) a Hash Code mapping layer (red layer); (iv) a novel pairwise loss function with binary constraints.

on classification such as image retrieval. To overcome this challenge, we automatically create a training set of matching and non-matching image pairs by exploiting 3D reconstructed models and their associated images. In particular, we make use of the 3D models given by [39], in which there are 713 3D models reconstructed from images downloaded from Flickr. Most of reconstructed models are popular landmarks and cities. The authors released 3D models and 30K images which were used to build models. $5,974$ and $1,691$ images are selected as training queries and validation queries, respectively.

In order to mining matching pairs, we follow the procedure used in [39]. Given a training query image and its 3D model membership, we select images that belong to the same 3D model and co-observe enough 3D points. Among these, one image is randomly sampled and used as the matching image for the query. The set of matching pairs is kept during the training. In order to mining non-matching pairs, different from [39], we perform two stages of generation. The offline stage generates pairs for training the network at first iterations. After a certain iterations, we use the current network to perform the online generation (i.e., regenerating non-matching pairs) and use new pairs to continue the training. In particular, the offline generation is performed as follows: Given a training query image, we select top k "nearest" images from 3D models different than the model containing the query. The distances between images are computed by using features extracted from the pre-trained network [39]. Among these k non-matching images, we randomly sample m images (with at most one image per model) as the non-matching ones for the query. After a certain of iterations, we perform the online non-matching pair generation. The online generation is similar to the offline ones, except that the distance between images is computed by using the binary codes generated by the hash code layer of the current network. The values of k and m are empirically set to 70 and 6 in our experiments. Note that our non-matching mining strategy is different from [39] at two main aspects. Firstly, in [39], the authors selected top m nearest ones (i.e., hardest negative) from k non-matching images. We empirically found that randomly selecting hard negative images gives better retrieval results than selecting the hardest ones. This is consistent with the observation in [43, 26], i.e., using the hardest negative samples can in practice lead to bad local minima in training. Secondly, because our target is to learn binary codes, hence in the online stage, we mine negative images using binary codes produced by the HC layer, rather than the features produced by the MAC layer as in [39].

3.3. Pairwise binary constrained loss

Given the training image pairs, we aim to train the network which not only produces binary codes but also ensures the discrimination of the codes, i.e., matching images should likely have similar binary codes, or vice versa. As the Hamming distance between two strings of binary codes is one-to-one corresponding to their Euclidean distance, we propose to minimize the following binary constrained loss function which acts on the pairs

$$\min_{\mathbf{W}} \mathcal{L}(i,j) = y_{ij} \left\|\mathbf{f}_i - \mathbf{f}_j\right\|_2 + (1 - y_{ij}) \max\left(0, c - \left\|\mathbf{f}_i - \mathbf{f}_j\right\|_2\right) \tag{2}$$

$$\text{s.t. } \mathbf{f}_i, \mathbf{f}_j \in \{-1, 1\}^L \tag{3}$$

where $\mathbf{W}$ is the network parameters; $\mathbf{f}_i$ and $\mathbf{f}_j$ are outputs of the Hash Code layer for input images i and j, respectively; the label $y_{ij} \in \{0, 1\}$ indicates that the image pair i, j is matching ($y_{ij} = 1$) or non-matching ($y_{ij} = 0$); L is the code length; c is a constant.

Technically, the loss function will encourage matching pairs to have similar hash codes, and non-matching pairs to have different hash codes. When a non-matching pair has a large enough distance, i.e. $\geq c$, it is not to be

taken into account in the loss. The constraint (3) is to ensure the network outputs are binary. Optimizing the loss function (2) with the binary constraint (3) using stochastic gradient method is difficult since the constraints are not differentiable. To overcome this challenge, we utilize the idea of the penalty method [35]. This leads to a formulation which avoids solving the exact binary constraint but instead minimizes the binary quantization loss. This makes sense because when the binary quantization loss approaches zero, the binary constraints are approximately satisfied. Specifically, we introduce new auxiliary binary variables $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^{N} \in \{-1, 1\}^{L \times N}$ where N is number of current training images, and minimize the following loss function

$$\min_{\mathbf{W},\mathbf{B}} \mathcal{L}(i,j) = y_{ij} \left\|\mathbf{f}_i - \mathbf{f}_j\right\|^2 + (1 - y_{ij}) \left(\max\left(0, c - \left\|\mathbf{f}_i - \mathbf{f}_j\right\|\right)\right)^2 + \alpha \left(\left\|\mathbf{f}_i - \mathbf{b}_i\right\|^2 + \left\|\mathbf{f}_j - \mathbf{b}_j\right\|^2\right) \quad (4)$$

$$\text{s.t. } \mathbf{b}_i, \mathbf{b}_j \in \{-1, 1\}^L \quad (5)$$

where α is a weighting parameter. The third term of (4) forces the output of the Hash Code layer (i.e., $\mathbf{f}_i, \mathbf{f}_j$) as close to binary values as possible, i.e., it minimizes the binary quantization loss. Although the new loss function still contains constraints, the variables $\mathbf{B}$ and $\mathbf{W}$ are decoupled. This allows us to apply alternating optimization over these variables. We will show shortly that $\mathbf{W}$ now can be optimized using the stochastic gradient decent method and $\mathbf{B}$ has a closed-form solution.

3.4. Parameter learning

In order to minimize the loss function (4) under the constraint (5), we propose an alternating optimization approach, i.e., we learn each variable (network parameter $\mathbf{W}$ or $\mathbf{B}$) at a time while holding the other fixed.

- Fix $\mathbf{W}$, solve $\mathbf{B}$: Let $\mathbf{F} = \{\mathbf{f}_i\}_{i=1}^{N}$, if $\mathbf{W}$ is fixed, the optimal solution for $\mathbf{B}$ is $sgn(\mathbf{F})$.
- Fix $\mathbf{B}$, solve $\mathbf{W}$: When $\mathbf{B}$ is fixed, the binary constraint (5) can be ignored, thus, the network parameters $\mathbf{W}$ can be optimized by minimizing the loss (4) using the standard back-propagation. A number of epochs is run until $\mathbf{W}$ converges to local minima before switching to $\mathbf{B}$.

The whole learning process is summarized in the Algorithm 1. In the Algorithm, $\mathbf{W}_{(t)}, \mathbf{F}_{(t)}, \mathbf{B}_{(t)}$ are values of $\mathbf{W}, \mathbf{F}, \mathbf{B}$ at t^{th} iteration. We implement the proposed approach using the MatConvNet toolbox [50]. At the begining (line 2 in the Algorithm), we initialize the feature extraction component with the pretrained MAC network [39] in which its loss layer is removed. The DR layer is initialized by using PCA weights on the training dataset with pretrained MAC features.

Algorithm 1 Parameter learning

Input:
Reconstructed 3D models and their images; L: code length; K: number of online non-matching pairs generation; T: number of iterations for training network, given a fixed set of matching / non-matching pairs.

Output:
Set of network parameters $\{\mathbf{W}\}$.

1: Offline generation of matching and non-matching pairs using the pretrained MAC network [39].
2: Initialize the network $\mathbf{W}_{(0)}$
3: **for** $k = 1 \rightarrow K$ **do**
4: **for** $t = 1 \rightarrow T$ **do**
5: Fix $\mathbf{W}_{(t-1)}$, compute $\mathbf{B}_{(t)} = sgn(\mathbf{F}_{(t-1)})$
6: Fix $\mathbf{B}_{(t)}$, optimize $\mathbf{W}_{(t)}$ (using $\mathbf{W}_{(t-1)}$ as initialization) using back-propagation. Save $\mathbf{W}_{(t)}$.
7: **end for**
8: Regenerate non-matching pairs using $\mathbf{W}_{(T)}$
9: Reinitialize $\mathbf{W}_{(0)} = \mathbf{W}_{(T)}$
10: **end for**

Inside the second *for loop*, we fix the training pairs and alternative solving $\mathbf{B}$ and $\mathbf{W}$. When fixing $\mathbf{B}_{(t)}$ and learning $\mathbf{W}_{(t)}$ (line 6 in the Algorithm), we train the network with a fix number of epochs np. The values of K, T and np are set to 4, 5 and 10, respectively. The values of c and α in (4) are set to $\frac{L}{2}$ and 1, respectively. The batch size for learning $\mathbf{W}_{(t)}$ is 28 pairs, i.e., 4 query images; each provides 1 matching pair and 6 non-matching pairs). Inside an iteration k, for each query, we generate $m = 6$ non-matching pairs (line 8 in the Algorithm). The best network is selected based on mean Average Precision (mAP) on the validation set.

4. Experiments

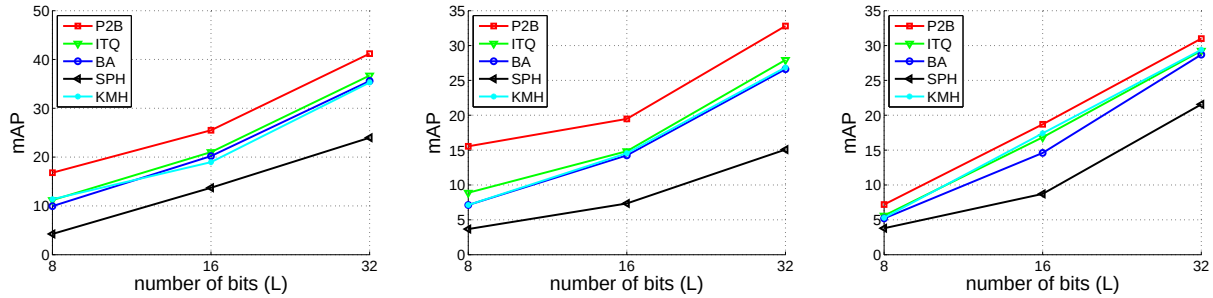
To evaluate the proposed method, which is dubbed as *P2B (Pixels to Binary codes)*, we conduct extensive image retrieval experiments on standard image retrieval benchmarks.

4.1. Dataset and baselines

Dataset We conduct experiments on Holidays [17], Oxford5k [37] and Oxford105k [37] datasets which are widely used in evaluating image retrieval systems [19, 2, 18].

Holidays The Holidays dataset consists of 1,491 images of different locations and objects, 500 of them being used as queries. Most of images in dataset are natural scenes. Follow the standard protocol [19, 2], when evaluating, we remove the query from the ranked list.

Oxford5k The Oxford5k dataset consists of 5,063 im-



(a) Oxford5k (b) Oxford105k (c) Holidays

Figure 2: mAP comparison between the proposed P2B and the traditional unsupervised hashing methods on Oxford5k, Oxford105k, and Holidays datasets.

ages of buildings and 55 query images corresponding to 11 distinct buildings in Oxford. Follow the standard protocol [19, 2], we crop the bounding boxes of the region of interest and use them as the queries.

Oxford105k In order to evaluate the proposed method at larger scale, we merge Oxford5k dataset with 100k distracted images downloaded from Flickr [37], forming the Oxford105k dataset.

The ground truth of queries have been provided with the datasets. Following previous works [6, 31, 48, 56], we evaluate the performance of methods at different code lengths, i.e. 8, 16, 32, 256, 512 bits. The retrieval accuracy is measured by mean Average Precision (mAP).

Baselines We compare the proposed P2B against state-of-the-art unsupervised hashing methods, including both traditional methods: Iterative Quantization (ITQ) [12], Binary Autoencoder (BA) [6], Spherical Hashing (SPH) [16], K-means Hashing (KMH) [15], and recent deep-based methods: DeepBit [31], Deep Region Hashing (DRH) [48].

We also compare P2B against supervised hashing methods, including traditional methods: Kernel-based Supervised Hashing (KSH) [33], Binary Reconstructive Embedding (BRE) [23], ITQ-CCA [12], and recent deep-based methods: Supervised Semantics-preserving Deep Hashing (SSDH) [56], Hierachical Deep Hashing (HDH) [47]. Note that, to the best of our knowledge, very few supervised hashing methods [56, 47] have evaluated on standard image retrieval datasets such as Holidays, Oxford5K, and Oxford105K. It may be because there are no available large scale labeled training data for those datasets.

Furthemore, we also compare P2B with the recent state-of-the-art real-valued image representations for the image retrieval problem. They are regional maximum activation of convolution (R-MAC) [49], triangulation embedding (T-emb) [19], function approximation-based embedding (F-FAemb) [7], off-the-shelf CNN (OS-CNN) [41], faster-RCNN [42], neural codes [5], sum pooling of convolutional feature (SPoC) [4], cross-dimensional weighting (CDW) [20], unsupervised fine-tuning CNN (UF-CNN) [39], regional attention based deep feature (RADF) [21], CNN-based VLAD (NetVLAD) [1], the end-to-end CNN (E2E-CNN) [13].

4.2. Comparison with unsupervised hashing methods

4.2.1 Comparison with traditional unsupervised hashing methods

All compared traditional unsupervised methods require image features as inputs, instead of raw images. In order to make a fair comparison, we use the pretrained network [39] to extract the MAC features of 30K images which are used to reconstruct 3D models and use them as training inputs for traditional unsupervised hashing methods.

Figure 2 shows the comparative retrieval results between methods in term of mAP. On the Oxford5k and Oxford105k datasets, the results show that the proposed P2B significantly outperforms other methods, i.e., P2B outperforms the most competitive ITQ [12] $\geq 4.5\%$ mAP at all code lengths. The improvements are clearer at the lower code lengths, i.e., at $L = 8$, P2B outperforms ITQ 5.6% and 6.7% mAP on Oxford5k and Oxford105k, respectively.

On the Holidays dataset, the proposed P2B also outperforms ITQ [12] and other methods, i.e., P2B outperforms ITQ around 1.5% to 2% mAP at different code lengths. The results from Figure 2 also show that the improvements of P2B over other methods are clearer on the Oxford5k and Oxford105k datasets than on the Holidays dataset. The possible reason is that the Oxford5k building dataset may share similar visual characteristics (e.g., man-made architecture) with the training images which mostly contain landmarks and buildings. These datasets have different characteristics with the Holidays dataset which mostly contains images of natural scenes.

L	256	512
DeepBit [31]	60.30	62.70
DRH [48]	58.30	66.80
P2B	69.20	74.84

Table 1: mAP results of P2B, DeepBit [31], DRH [48] on Oxford5k. The results of the compared methods are cited from the corresponding papers.

L	256	512
SSDH [56]	-	63.80
HDH [47]	69.70	70.50
P2B	69.20	74.84

Table 2: Comparative mAP between the proposed P2B and recent supervised deep hashing methods on the Oxford5k dataset. The results of the compared methods are cited from the corresponding papers.

4.2.2 Comparison with unsupervised deep hashing methods

Here we compare the proposed P2B with the DeepBit [31] and Deep Region Hashing (DRH) [48], which are the state-of-the-art deep learning-based unsupervised hashing methods. To make a fair comparison to DRH, we compare to its results at comparable code lengths (e.g., $L = 256, 512$) without query expansion[1]. Note that, in DRH the authors used a part of Oxford5k dataset to train their hashing layer which is a non-standard training setting when evaluating on Oxford5k [19, 2]. In DeepBit [31], the authors trained their model using the semi-manually labeled landmark dataset [5] which is expected to have less noise level than our automatically created training dataset. The results from the Table 1 show that, at the comparable code lengths, the proposed P2B significantly outperforms the compared methods. P2B outperforms DeepBit 12.1% and outperforms DRH 8% mAP at $L = 512$, even when DRH is trained on images from the Oxford5k dataset and DeepBit is trained on the manually labeled landmark dataset.

4.3. Comparison with supervised hashing methods

4.3.1 Comparison with traditional supervised hashing methods

The supervised hashing methods, such as KSH, BRE, CCA-ITQ, require the class label to perform the training. To make a fair comparison, we use the images which are used to reconstruct the 3D models and carefully investigate different strategies to define the sample similarity when training traditional supervised hashing methods.

As the number of images of each 3D model varies, i.e., the largest model contains 80 images, while the smallest model contains only 23 images, we first select top 100 biggest 3D models and then randomly select 60 images per model for training. Note that as KSH and BRE require the full similarity between every pair of samples when training, it is difficult for these methods to handle larger training data. Let the similarity matrix be S, we try two approaches to define the similarity for each image pair in the matrix S.

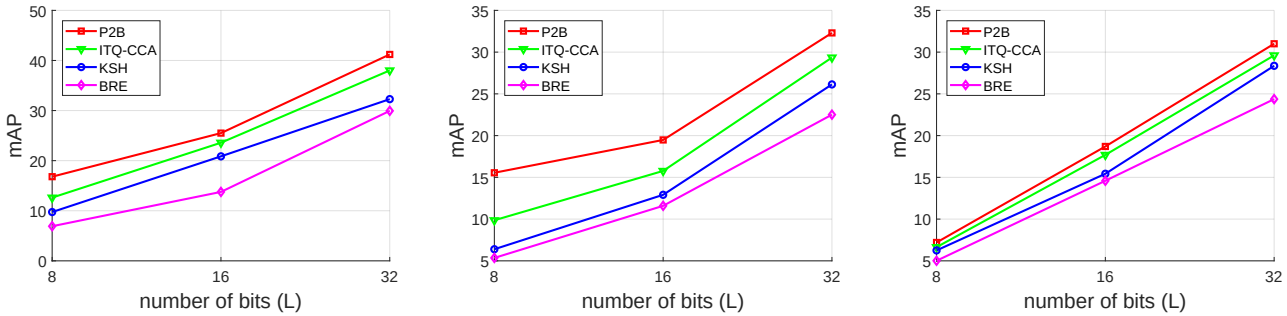
In the first approach, we check every image pair in the matrix S and use the same matching pair generation strategy in Section 3.2 for determining matching pairs. A pair (i, j) is matched, i.e. $S(i, j) = 1$, if both images belong to the same 3D model and co-observe enough 3D points, otherwise $S(i, j) = 0$. By using this first approach, we found that the matrix S is very sparse. In the second approach, images which are belonged to the same model are considered as matching, otherwise, non matching. In the other words, each 3D model is considered as a class. We empirically found that for KSH and BRE, the similarity matrix constructed by the second approach gives better retrieval results than the first approach, e.g., for KSH on Oxford5k dataset, the mAP at $L = 32$ is 19.11% and 32.28% for the first and the second approach, respectively. We found that although the first approach uses better strategy for creating matching pairs, the matrix S is very sparse. Hence the non-matching pairs strongly dominate the matching pairs during training, leading to poor results. In the following, we consistently use the second approach, which considers each 3D model as a class, for training the compared methods.

Figure 3 presents the comparative retrieval results of P2B, KSH, BRE, CCA-ITQ. The results show that on the Oxford5k and Oxford105 datasets, P2B outperforms compared methods with fair margin at all code lengths. On the Holidays dataset, P2B and CCA-ITQ achieve comparable results and these methods outperform KSH and BRE.

4.3.2 Comparison with supervised deep hashing methods

There are very few deep supervised hashing methods that report results on image retrieval benchmarks such as Holidays and Oxford5k. In this section we compare the proposed P2B to the recent deep supervised hashing methods SSDH [56], hierarchical deep hashing (HDH) [47] which have reported their results on Oxford5k dataset. Table 2 presents the comparative mAP between methods. It is worth mentioning that in SSDH [56], the authors fine-tuned their model on a semi-manually labelled landmark dataset [5], while in HDH [47], the authors fine-tuned their model using a part of the Oxford5k dataset, which is a non-standard training setting [19, 2]. Diffrent from those works, our

[1] In DRH [48] the authors achieved best mAP at 85.1% with multi-stage searching and query expansion at code length $L = 4096$.



(a) Oxford5k (b) Oxford105k (c) Holidays

Figure 3: mAP comparison between the proposed P2B and the traditional supervised hashing methods on Oxford5k, Oxford105k, and Holidays datasets.

method uses automatically created training data from unlabeled images. The results in Table 2 show that P2B outperforms SSDH a large margin, i.e., 11% mAP at $L = 512$. Compare to HDH, P2B and HDH achieve comparable mAP at $L = 256$, while at higher code length, i.e. $L = 512$, P2B improves 4.3% mAP over HDH. It is also worth mentioning that our automatically created training data has higher noise level than the semi-manually labeled dataset used in SSDH. Specifically, we fine-tuned SSDH with the images used to reconstruct 3D models, i.e., we select top 100 biggest 3D models and consider each 3D model as a class. The mAP of the fine-tuned SSDH using these images at $L = 512$ is 59.17% which is lower than the reported 63.80% in [56]. Inspite of the higher noise level in the training data, P2B achieves a mAP 74.84% which is higher than the reported 63.80% of SSDH. This confirms the effectiveness of the proposed framework.

4.4. Comparison with real-valued image representations

In this section we compare the proposed P2B to the state-of-the-art real-valued image representations. Table 3 presents the comparative mAP between methods on Oxford5k dataset. The results show that, even using binary representation, P2B outperforms most compared methods that use the real-valued representations. The state-of-the-art mAP is achieved by the real-valued high dimensional representation E2E-CNN [13]. In particular, E2E-CNN outperforms P2B 11.3% mAP. Although the proposed P2B does not provide the state-of-the-art retrieval accuracy, it is worth mentioning that in term of memory, P2B uses only 64 bytes to represent an image, which is 128 times less than the one of E2E-CNN. Furthemore, the binary representation of P2B also allows the fast distance calculation, i.e., Hamming distance. This makes P2B more suitable than E2E-CNN for the large scale retrieval problem.

Methods	D	mAP
R-MAC [49]	512 (float)	66.9
T-emb [19]	1024 (float)	56.2
F-FAemb [7]	1024 (float)	58.2
OS-CNN [41]	4096 (float)	68.0
Faster-RCNN [42]	4096 (float)	67.8
Neural codes [5]	256 (float)	55.7
SPoC [4]	256 (float)	58.9
CDW [20]	256 (float)	65.4
NetVLAD [1]	256 (float)	63.5
UF-CNN [39]	512 (float)	79.7
RADF [21]	2048 (float)	76.8
E2E-CNN [13]	2048 (float)	86.1
P2B	512 (bit)	74.8

Table 3: Comparative mAP between P2B and the state-of-the-art real-valued representations on Oxford5k dataset. The second column is dimensionality of the presentations. Note that the compared methods uses real-valued ($float$) representations, while P2B uses binary (bit) presentations.

5. Conclusion

In this paper, we propose a novel end-to-end deep hashing framework for directly learning compact binary codes from images without the need of manual annotation. We exploit the reconstructed 3D models and their associated images to automatically create the training data. We propose a novel pairwise binary constrained loss function which not only allows to leverage the discriminative information from training data but also encourages the output codes to be binary. We also propose an efficient alternating optimization to train the network under the constrained loss. The solid experimental results on image retrieval benchmark datasets show that the proposed method compares favorably with the state of the art.

References

[1] R. Arandjelovic, P. Gronát, A. Torii, T. Pajdla, and J. Sivic. Netvlad: CNN architecture for weakly supervised place recognition. *TPAMI*, pages 1437–1451, 2018.

[2] R. Arandjelovic and A. Zisserman. All about VLAD. In *CVPR*, 2013.

[3] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson. From generic to specific deep representations for visual recognition. In *CVPRW*, 2015.

[4] A. Babenko and V. S. Lempitsky. Aggregating local deep features for image retrieval. In *ICCV*, 2015.

[5] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky. Neural codes for image retrieval. In *ECCV*, 2014.

[6] M. A. Carreira-Perpinan and R. Raziperchikolaei. Hashing with binary autoencoders. In *CVPR*, 2015.

[7] T.-T. Do and N.-M. Cheung. Embedding based on function approximation for large scale image search. *TPAMI*, pages 626–638, 2018.

[8] T.-T. Do, A.-D. Doan, and N.-M. Cheung. Learning to hash with binary deep neural network. In *ECCV*, 2016.

[9] S. Ercoli, M. Bertini, and A. D. Bimbo. Compact hash codes for efficient visual descriptors retrieval in large scale databases. *TMM*, pages 2521–2532, 2017.

[10] O. C. Filip Radenovic, Giorgos Tolias. Fine-tuning cnn image retrieval with no human annotation. *TPAMI*, 2018.

[11] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, 1999.

[12] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, 2011.

[13] A. Gordo, J. Almazán, J. Revaud, and D. Larlus. End-to-end learning of deep visual representations for image retrieval. *IJCV*, pages 237–254, 2017.

[14] K. Grauman and R. Fergus. Learning binary hash codes for large-scale image search. *Machine Learning for Computer Vision*, 2013.

[15] K. He, F. Wen, and J. Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *CVPR*, 2013.

[16] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-e. Yoon. Spherical hashing. In *CVPR*, 2012.

[17] H. Jégou, M. Douze, and C. Schmid. Improving bag-of-features for large scale image search. *IJCV*, pages 316–336, 2010.

[18] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[19] H. Jégou and A. Zisserman. Triangulation embedding and democratic aggregation for image search. In *CVPR*, 2014.

[20] Y. Kalantidis, C. Mellina, and S. Osindero. Cross-dimensional weighting for aggregated deep convolutional features. In *ECCV Workshops*, 2016.

[21] J. Kim and S.-E. Yoon. Regional attention based deep feature for image retrieval. In *British Machine Vision Conference (BMVC)*, 2018.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[23] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, 2009.

[24] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, 2009.

[25] B. Kulis, P. Jain, and K. Grauman. Fast similarity search for learned metrics. *TPAMI*, pages 2143–2157, 2009.

[26] B. G. V. Kumar, B. Harwood, G. Carneiro, I. D. Reid, and T. Drummond. Smart mining for deep metric learning. In *ICCV*, 2017.

[27] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*, 2015.

[28] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, pages 2278–2324, 1998.

[29] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In *CVPR*, 2014.

[30] K. Lin, J. Lu, C.-S. Chen, and J. Zhou. Learning compact binary descriptors with unsupervised deep neural networks. In *CVPR*, 2016.

[31] K. Lin, J. Lu, C.-S. Chen, J. Zhou, and M.-T. Sun. Unsupervised deep learning of compact binary descriptors. *TPAMI*, 2018.

[32] H. Liu, R. Wang, S. Shan, and X. Chen. Deep supervised hashing for fast image retrieval. In *CVPR*, 2016.

[33] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, 2012.

[34] J. Lu, V. E. Liong, and J. Zhou. Deep hashing for scalable image search. *TIP*, 2017.

[35] J. Nocedal and S. J. Wright. *Numerical Optimization*. World Scientific, 2nd edition, 2006.

[36] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *ICML*, 2011.

[37] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

[38] F. Radenovic, J. L. Schönberger, D. Ji, J. Frahm, O. Chum, and J. Matas. From dusk till dawn: Modeling in the dark. In *CVPR*, 2016.

[39] F. Radenovic, G. Tolias, and O. Chum. CNN image retrieval learns from bow: Unsupervised fine-tuning with hard examples. In *ECCV*, 2016.

[40] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*, 2009.

[41] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *CVPRW*, 2014.

[42] A. Salvador, X. Giró-i-Nieto, F. Marqués, and S. Satoh. Faster R-CNN features for instance search. In *CVPR Workshops*, 2016.

[43] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.

[44] F. Shen, C. Shen, W. Liu, and H. Tao Shen. Supervised discrete hashing. In *CVPR*, 2015.

[45] F. Shen, Y. Xu, L. Liu, Y. Yang, Z. Huang, and H. T. Shen. Unsupervised deep hashing with similarity-adaptive and discrete optimization. *TPAMI*, 2018.

[46] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014.

[47] G. Song and X. Tan. Hierarchical deep hashing for image retrieval. *Frontiers of Computer Science, Springer*, pages 253–265, 2017.

[48] J. Song, T. He, L. Gao, X. Xu, and H. T. Shen. Deep region hashing for generic instance search from images. In *AAAI*, 2018.

[49] G. Tolias, R. Sicre, and H. Jégou. Particular object retrieval with integral max-pooling of CNN activations. In *ICLR*, 2016.

[50] A. Vedaldi and K. Lenc. Matconvnet - convolutional neural networks for MATLAB. *CoRR*, 2014.

[51] J. Wang, S. Kumar, and S. Chang. Semi-supervised hashing for large-scale search. *TPAMI*, pages 2393–2406, 2012.

[52] J. Wang, W. Liu, S. Kumar, and S. Chang. Learning to hash for indexing big data - A survey. *Proceedings of the IEEE*, 2015.

[53] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen. A survey on learning to hash. *TPAMI*, 2017.

[54] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008.

[55] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI*, 2014.

[56] H.-F. Yang, K. Lin, and C.-S. Chen. Supervised learning of semantics-preserving hash via deep convolutional neural networks. *TPAMI*, pages 437–451, 2018.

[57] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. *IEEE TIP*, pages 4766–4779, 2015.

[58] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *CVPR*, 2015.

[59] B. Zhuang, G. Lin, C. Shen, and I. D. Reid. Fast training of triplet-based deep binary embedding networks. In *CVPR*, 2016.